



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/028,833	12/20/2001	Matthew W. Weismiller	8266-0685	4403

7590 05/17/2004

Timothy E. Niednagel
Bose McKinney & Evans LLP
Suite 2700
135 N. Pennsylvania Street
Indianapolis, IN 46204

EXAMINER

TRETTEL, MICHAEL

ART UNIT	PAPER NUMBER
----------	--------------

3673

DATE MAILED: 05/17/2004

Please find below and/or attached an Office communication concerning this application or proceeding.



UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
P.O. Box 1450
ALEXANDRIA, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES**

Paper No. 20

Application Number: 10/028,833
Filing Date: December 20, 2001
Appellant(s): WEISMILLER ET AL.

Timothy Niednagel
For Appellant

MAILED

MAY 17 2004

GROUP 3600

EXAMINER'S ANSWER

This is in response to the appeal brief filed February 6, 2004.

(1) *Real Party in Interest*

A statement identifying the real party in interest is contained in the brief.

(2) *Related Appeals and Interferences*

A statement identifying the related appeals and interferences which will directly affect or be directly affected by or have a bearing on the decision in the pending appeal is contained in the brief.

(3) *Status of Claims*

The statement of the status of the claims contained in the brief is correct.

(4) *Status of Amendments After Final*

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) *Summary of Invention*

The summary of invention contained in the brief is correct.

(6) *Issues*

The appellant's statement of the issues in the brief is correct.

(7) *Grouping of Claims*

Appellant's brief includes a statement that claims 38, 40 to 44, and 65 to 69 do not stand or fall together and provides reasons as set forth in 37 CFR 1.192(c)(7) and (c)(8). For the purposes of appeal claims 38, 41, and 68 have been grouped together, with the remaining claims being grouped separately.

(8) *Claims Appealed*

The copy of the appealed claims contained in the Appendix to the brief is correct.

(9) Prior Art of Record

5,097,550	Marra, Jr.	03-1992
4,612,679	Mitchell	09-1986
5,542,138	Williams et al)8-1996

(10) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 38 and 65 to 69 are rejected under 35 U.S.C. 103(a) as being unpatentable over Marra, Jr. in view of Williams et al. This rejection is set forth in prior Office Action, Paper No. 15.

Claims 38, 40 to 44, 65, 68, and 69 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mitchell in view of Williams et al. This rejection is set forth in prior Office Action, Paper No. 15.

(11) Response to Argument

The appellant's argument with regards to the §103(a) rejection over the combination of Marra and Williams is misleading. Specifically, the appellant has asserted on page 4 of the brief that the combination would lead to only the inclusion of the processor 82 of the Williams patent, while leaving the housings 32, 34 of the Marra patent in place as the equivalent of the claimed display screen. This is incorrect, the examiner believed that it was clear from the rejection that the skilled artisan would replace the control elements in the housings 32, 34 of the Marra sideguard with the equivalent display screen 70, keyboard 72 and pointing device 74, and control

Art Unit: 3673

panel 58 of the Williams patent. For the proposed combination to work effectively it would be necessary to do so because the housings 32, 34 of the Marra sideguard contain simple switches and indicator lights used as part of the included control system. It would not make any sense whatsoever to include a processor control such as the one taught by Williams in the Marra sideguard without also including the graphical display screen as a means for operating, controlling, and interfacing with the processor control along with the other means (keyboard, pointing device, etc.) used to interface with and control the processor.

The appellant's arguments against the motivation for combining the Williams and Marra references considers the references singly without considering the prior art as a whole. In response to appellant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986). In the present case the motivation for combining the references is rather simple-the skilled artisan would recognize that a processor type control for a bed such as the one taught by Williams would inherently be superior to an older analog/manual type control for a bed such as the one shown by Marra. As shown by Marra (and also Mitchell) it is already well known in the art to integrate bed controls into a bed sideguard. In essence, the examiner is proposing that the skilled artisan would upgrade the control system used in the Marra sideguard with the more flexible and far more useful type of control offered by a processor based control such as the one taught by Williams. The reasons pointed out by the appellant that allegedly teach away from the combination in fact do no such thing, instead they only discuss the differences and alleged superiority of any type of

Art Unit: 3673

bed control system when mounted upon a pendant as used by Williams. This does not mean that the skilled artisan would then simply stop using sideguard mounted type bed control systems in favor of only a pendant mounted system, instead this means that a pendant mounted control system has its own peculiar advantages (and also disadvantages) over a sideguard mounted control system under some circumstances. However, the control systems themselves are still equivalents within the art since they serve a similar function (controlling the operation of a hospital bed) within a similar environment (the environment of the bed). It is presumed that the skilled artisan would be aware of these differing types of control systems known within this common body of art, including the advantages and disadvantages of each system. Once a better type of control system is known within the art the advantages of using it in an older type of integrated control would be natural and expected.

The argument made concerning claim 65 has been noted. The appellant should read column 4, lines 17 to 34 of Williams, and in particular line 27. Williams discloses that the control module 40 includes a memory device 90 that can be used for patient data charting. The data can be stored in the control module 40 by means of being input through the keyboard 72 and stored in the memory 90. Instructions of use of the control module 40 can also be stored in the memory 90 for use as a help system. These instructions can be recalled with suitable prompting messages and output upon the display 70. Claim 65 only specifies one of a list of possible displayed graphical information, including "a bed status". Since the control module 40 is used to control the bed and includes an integrated help system that outputs to the display the examiner submits that the help system would include messages that inform a user on how to use the bed control module 40 and/or how to operate the bed with the control module 40. In addition patient

Art Unit: 3673

charting data is stored in the memory 90, and it would appear to be an inherent feature of the Williams device to output this data upon the display as needed-otherwise it would appear that the display serves little purpose.

As regards claim 66 the appellant should consult column 4, lines 28 to 31 of Williams. Disclosure is made of the use of a modem 97, and infrared transmitter/receiver link 96, or other data transmitting means. Since the use of wired data networks is so very well known as to have completely saturated society at this point the examiner submits that the skilled artisan would consult and/or make use of the well known technical features of wired and remote networking as commonly used in the computing industry in view of this suggestion.

The argument made concerning claim 67 has been noted. The examiner has provided copies of publicly available documentation as an attachment to the Examiner's Answer that supports the examiner's contention that the concept of using a remote display over a network in order to control a processor was old and well known in the art at the time the invention was first made. Since the examiner is familiar with the use of the X Window system as a means for controlling remote networked computers the documents have been drawn from <http://www.x.org>, and also from the description provided by the Wikipedia online encyclopedia at http://en.wikipedia.org/wiki/X_Window_System. These documents show that this type of remote display and control of a network connected computer dates from at least 1984, in the examiner's case personal use was made of X Window terminals in the USPTO Scientific Library in the early 1990's.

The argument made concerning claim 69 has been noted. Claim 69 is similar to claim 65 in that it sets forth a selection of at least one item from a list of graphical elements. As set forth

Art Unit: 3673

above in the argument made concerning claim 65, the appellant should consult column 4, lines 17 to 34 of Williams. Merely operating the control module 40 in order to display the integrated help system upon the display panel 70 is enough to meet the terms of the claim, since the help system can be considered a “user selectable function”.

The argument made against the combination of Mitchell and Williams as applied against claims 38, 41, and 68 is unpersuasive for the same reasons applied above when considering the argument made against the combination of Marra, Jr. and Williams. In addition the appellant should note that it is not necessary for Mitchell to include a “display screen” as alleged in the argument, because the use of a display screen as part of a control system is taught by Williams. This is a feature that would be added by the combination of references, and is not needed to be present upon Mitchell in the first place.

As regards claim 40, the examiner notes that Williams includes the features alleged to be missing from the rejection. In particular Williams has a control panel 52 that includes many user operable switches in the form of buttons 54, 56, and 58, and additionally includes a keyboard 72. Any computer keyboard is in fact a series of user operable switches, since they operate by selectively interrupting a switch element engaged by pressing the key.

As regards claim 42, the examiner notes that the Mitchell sideguard includes a control panel 160 mounted upon an upper section 120 which is pivotally mounted to the sideguard frame 100 upon a set of bushings 114. This allows the upper section to be flipped relative to the bedframe, such that the control panel 160 can face inwardly or outwardly relative to the interior portion of the bed. See Figure 1 for an illustration of both positions. If the sideguard controls were replaced or upgraded as proposed by the examiner, the control panel display would then be

Art Unit: 3673

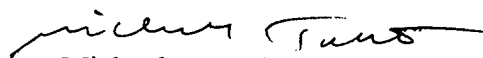
capable of being flipped to face either the interior or exterior of the bed as desired by the operator. Therefore the terms of claim 42 have been met.

As regards claim 43, please note that Williams includes a user input in the form of keyboard 72, a controller in electrical communication with the user input and display screen in the form of the processor and bus schematically shown in Figure 4, with the controller being configured to display variable graphical information on the display (column 4, lines 17 to 34).

The terms of claim 44 have been adequately met by the disclosure present in column 4, lines 17 to 34 of Williams. See also column 3, lines 33 to 47, and in particular line 45. Reference is made to the use of the keyboard 72 and trackball 74 as a means for inputting commands and data to the control module, the examiner submits that the only logical interpretation of this statement is that the control module uses a menu driven display system that allows such operation.

The arguments made concerning the rejection of claims 65 to 67 and 69 as being obvious over Mitchell in view of Williams have already been addressed above in detail. In short, the examiner submits that the elements present in these particular claims are already shown and disclosed in the Williams patent, and need not be present in the Mitchell patent considered by itself.

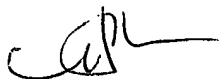
Art Unit: 3673



Michael Trettel
Primary Examiner
Art Unit 3673

Conferees

Heather Shackelford



Tom Will



X Window System

From Wikipedia, the free encyclopedia.

The **X Window System** (commonly **X11** or **X**) is a windowing system for computers with bitmap displays. It is standard on Unix, Linux and other Unix-like operating systems and is available for most other modern operating systems.

X provides basic underpinnings for a graphical user interface (GUI): drawing and moving windows on a screen and interacting with a mouse and a keyboard. As per its maxim of "mechanism, not policy," X does not specify or provide the user interface itself — this is handled by user software. As such, the visual interface of X-based environments is highly variable; several different interface styles may coexist on one computer.

X features *network transparency*: the machine where an application program runs (the application *client*) need not be the user's local machine (the display *server*). X's usage of the terms "client" and "server" is the reverse of what people often expect, in that the "server" is the user's local display rather than the remote machine.

X originated at MIT in 1984. The current version, X11, was released in 1987. The project is now led by the X.Org Foundation. The current reference release is version 11 release 6.7.0 (X11R6.7.0).

Table of contents [hide]

1 Architecture

1.1 X Display Managers

1.2 Services provided by X

1.2.1 X protocol interactions

1.3 Widget toolkits and desktop environments

1.4 Color modes of the X Window System

2 Implementations

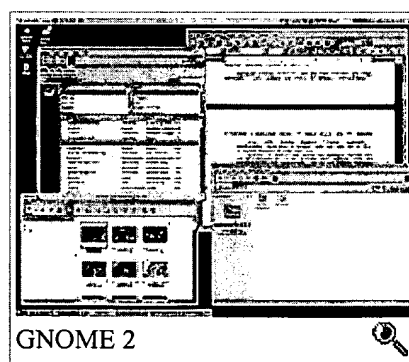
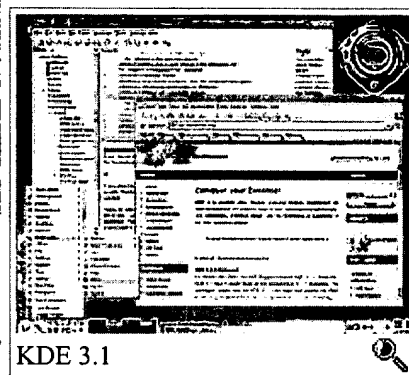
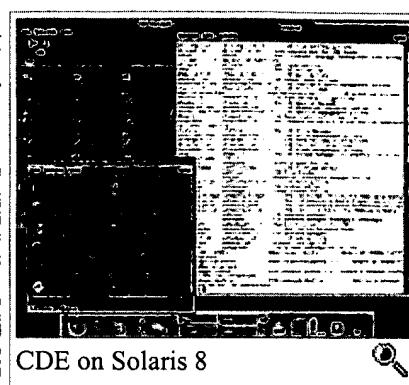
2.1 X terminals

3 History

4 Nomenclature

5 See also

6 External links



Architecture

X is based on a client-server model. A *display server* program runs on a computer with a graphical display and communicates with various *client programs*, accepting requests for graphical output (windows) and sending back user input (keyboard, mouse).

This client-server terminology — your terminal is the "server", the remote application is the "client" — often confuses new X users, because the terms appear reversed. But X takes the perspective of the program, rather than the end-user or the hardware: the remote programs connect to the X server display running on the local machine, and thus act as clients; the local X display accepts incoming traffic, and thus acts as a server.

The communication protocol between server and client runs network-transparently: the client and server may run on the same machine or on different ones, possibly with different architectures and operating systems. For example, one could run a large computational simulation on a far-away Unix supercomputer (the "client") while displaying the user interface and results on the local desktop machine (the "server", which could be running Windows or any other operating system). Other possibilities include administering a remote machine using graphical software on that machine, using remote application software from thin client machines with little or no processing power or storage, and running graphical software on several machines at once via a single display.

X Display Managers

The X Display Manager keeps the X server process alive on the X server machine, connecting it to a physical screen and serving a login prompt on this screen. XDM serves as the default display manager for X, but other projects have developed their own display managers:

- XDM
- GDM (developed by GNOME)
- KDM (developed by KDE)
- wdm (using the WINGs widget set used in Window Maker)
- entrance (using the architecture used in Enlightenment v.1.7)

Services provided by X

The X server provides only the following. All else is provided by client programs.

- **Input handling:** Keyboard and mouse input are passed to clients as events via the window manager — which, as far as X is concerned, is just another client.
- **Window services:** Clients ask the server to create or destroy windows. Windows may be nested hierarchically. Clients can request information about windows.
- **Text and fonts:** Clients ask for text to be drawn at a given location in a given font. The client can request information on available fonts.
- **Graphics:** Clients ask the server to draw pixels, lines or shapes or perform bitmap operations.

X protocol interactions

There are four types of communication between X clients and the server:

1. **Request:** The client requests an action or information.
2. **Reply:** The server responds. Note that not all requests generate replies.
3. **Event:** The server sends an event to the client, *e.g.* keyboard or mouse input, or a window being moved, resized or exposed.
4. **Error:** The server sends an error packet if a request is invalid. Note that requests are queued, so an error packet may not be sent immediately.

Widget toolkits and desktop environments

X provides "mechanism, not policy." As such, it does not specify or provide the user interface features, such as buttons, menus, window title bars and so on. These are provided by user software, such as window managers, GUI widget toolkits and desktop environments.

Early GUI toolkits for X included:

- Xaw (the Athena Widget Set)

- OLIT (OPEN LOOK Intrinsics Toolkit)
- XView
- Motif
- Tk

OLIT and XView function as the base toolkits for AT&T and Sun's OPEN LOOK GUI.

Motif provides the base toolkit for the Common Desktop Environment (CDE), which is the standard desktop environment used on commercial Unix systems such as Solaris and HP-UX. (GNOME is offered in Solaris 9 and will be standard in future versions.)

Within the last five years, other toolkits have grabbed developer and user mind share. The modern toolkits include:

- Qt (used by KDE, developed by TrollTech)
- GTK+ (GIMP Tool Kit, used by GNOME)
- wxWidgets
- fltk
- FOX

The KDE and GNOME desktop environments provide much better application functionality and services than is offered by plain window managers or older desktop environments.

Color modes of the X Window System

The colors used in X Window Systems sometimes confuse users, as old or special-purpose applications may require a certain color mode. Most modern applications use a color mode called "TrueColor", but historically X has supported several different modes:

- DirectColor
- GrayScale
- PseudoColor
- StaticColor
- StaticGray
- TrueColor

Implementations

X.Org distributes the X Window System at no charge, with source code included and no restrictions on modification or redistribution, under the MIT License and similar licenses.

Due to the liberal licensing, a number of implementations (both free and proprietary) have appeared, based on the code from MIT. Originally developed for the UNIX graphical workstations of the 1980s as part of MIT's Project

Athena, these enhanced versions mainly added compatibility with specific operating systems and hardware.

Commercial Unix vendors tend to take the X.org reference implementation and adapt it for their hardware, usually customising it heavily and adding proprietary extensions. One commercial implementation not tied to a hardware vendor is Accelerated-X.

The X variant most common on free Unix-like systems is XFree86. This originated from the X386 server included with the reference implementation, hence the name. Other open source X servers include the freedesktop.org Xserver project (led by Keith Packard, recently of XFree86) and the current X.Org reference implementation, the XOrg Foundation Open Source Public Implementation of X11, a fork of XFree86.

While Unix has standardized on X, X servers also exist for platforms with their own graphical environments, like Windows, Mac OS/MacOS X and OS/2. X servers used on Windows include Cygwin/X, Exceed, XVision, X-Win32 and WeirdX. Mac OS X 10.3 (Panther) release includes Apple's version of X11.

X terminals

An **X terminal** is a piece of dedicated hardware running an X server as a thin client. This architecture became popular for building inexpensive client parks for many users to simultaneously use the same large server. This use very much aligns with the original intention of the MIT project.

X terminals explore the available hosts using an *ad hoc* network protocol called XDMCP to connect to a specific server which presents a list of available hosts. This server in turn may gather a list of available hosts using broadcast on the local network.

Dedicated X terminals are no longer common; this functionality is now typically provided either with a user's Microsoft Windows PC running an X server program or a low-end PC running Linux.

History

X derives its name as a successor to a pre-1983 window system called W (the letter X directly following W in the Latin alphabet).

The X Window System concept first emerged in 1984, at the Massachusetts Institute of Technology (MIT), as a joint project between their Laboratory for Computer Science and Digital Equipment Corporation. The initial impetus came from MIT's Project Athena, which sought to provide easy access to computing resources for all students. Because MIT could not buy all the workstations needed, and no single vendor appeared willing to donate them, the Institute needed a platform-independent graphics system to link together its heterogeneous systems.

The first version to achieve wide deployment was Version 10 (X10) in 1986. Version 11 (X11), the current version, was released in 1987.

In 1988, a non-profit group called the (MIT) X Consortium formed to direct future development of X standards in an atmosphere inclusive of commercial and educational interests. The X Consortium produced several significant revisions to X11, the last being Release 6 in 1994 (X11R6). The Consortium dissolved at the end of 1996, producing a final, small revision called X11R6.3.

Stewardship of X then passed to The Open Group, an outgrowth of the Open Software Foundation (OSF), who produced the popular Motif widget toolkit for X. In early 1998, the Open Group released a further revision to X11R6, called X11R6.4. A departure from the traditional licensing terms, however, inhibited many vendors, including the XFree86 Project, Inc., from adopting this version of the X Window System. As a result, the Open Group relicensed X11R6.4 under terms identical with the traditional license in late 1998.

In May 1999, stewardship of the X Window System passed from the Open Group to X.Org, a non-profit organization focused exclusively on maintenance and further development of the X Window System. X.Org supervised the release of versions X11R6.5.1 onward.

Nomenclature

"X Window System" is commonly shortened to "X11" or simply "X." The term "X Windows" (in the manner of "Microsoft Windows") is officially deprecated and generally considered incorrect, though it has been in common use since the inception of X and has been used deliberately for literary effect, *e.g.* in the UNIX-HATERS Handbook.

See also

- X11 color names
- History of the graphical user interface

External links

- X.Org Foundation (official home page)
- An Introduction to X11 User Interfaces
- Introduction to X Windows

Edit this page | Discuss this page | Page history | What links here | Related changes

Other languages: Deutsch | Français | 日本語 | Nederlands | Polski | Svenska

Main Page | About Wikipedia | Recent changes |

This page was last modified 01:05, 2 May 2004. All text is available under the terms of the GNU Free Documentation License (see Copvrights for details). Disclaimers. Wikipedia is powered by



The X Protocol

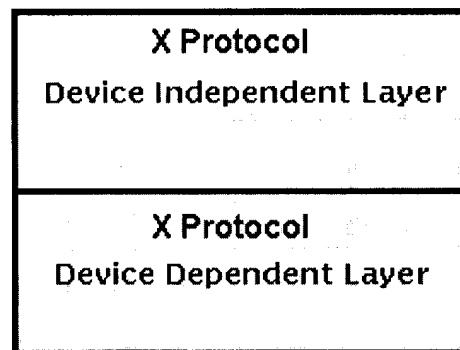


[X Protocol](#)
[Client-Server](#)
[Design](#)
[X Server Design](#)
[The Evolution of](#)
[Connectivity](#)
[Latest X Release](#)
[X Resources](#)
[Affiliated](#)
[Technologies](#)



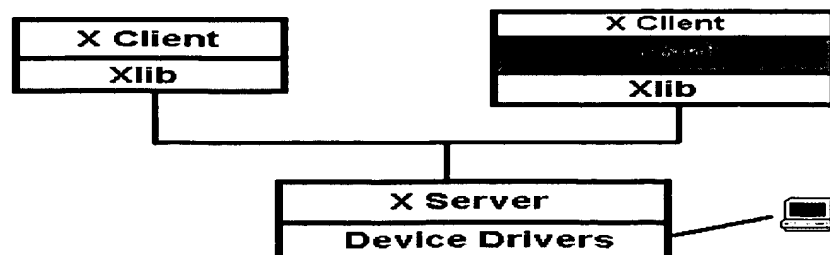
The X Protocol was developed in the mid 1980's amid the need to provide a network transparent graphical user interface primarily for the UNIX operating system. X provides for the display and management of graphical information, much in the same manner as Microsoft's Windows and IBM's Presentation Manager.

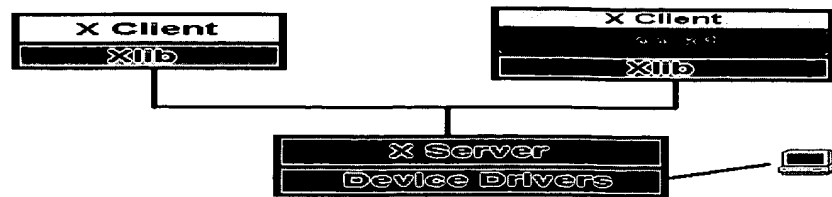
The key difference is in the structure of the X Protocol. Whereas Windows and Presentation Manager simply display graphical applications local to the PC, the X Protocol distributes the processing of applications by specifying a client-server relationship at the application level. The what to do part of the application is called an X client and is separated from the how to do part, the display, called the X server. X clients typically run on a remote machine which has excess computing power and displays on an X server. The benefit is true client-server and distributed processing.



Definition

The X Protocol defines a client-server relationship between an application and its display. To meet this the application (called an X client) is divorced from the display (known as the X server). X further provides a common windowing system by specifying both a device dependent and an independent layer, and basing the protocol on an asynchronous network protocol for communication between an X client and X server. In effect, the X Protocol hides the peculiarities of the operating system and the underlying hardware. This masking of architectural and engineering differences simplifies X client development and provides the springboard for the X Window System's high portability.





The advantages of this approach are many:

- Local and network based computing look and feel the same to both the user and the developer.
- The X server is highly portable allowing support for a variety of languages and operating systems.
- X clients also have a high degree of portability.
- X can support any byte stream oriented network protocol, local or remote.
- Applications do not suffer a performance penalty.



X Server Design



The design of an X server depends greatly upon the platform (hardware) and operating system on which it is implemented. As the capabilities of the underlying technologies increases, the power and capability of the X server also increases.



[X Protocol](#)
[Client-Server](#)
[Design](#)
[X Server Design](#)
[The Evolution of](#)
[Connectivity](#)
[Latest X Release](#)
[X Resources](#)
[Affiliated](#)
[Technologies](#)



Device Dependent Layer

- It is this layer that is responsible for localizing the X server to the native environment, be it Windows NT or Solaris.
- This layer swaps bytes of data from machines with differing byte ordering. Byte ordering (MSB and LSB) is noted in each X request.
- This layer hides the architectural differences in hardware and operating systems.
- Maintains device driver dependencies for keyboard, mouse and video.

Environment Architectures

- **Single Threaded Architecture** - The X server is a single sequential process using the native time-slice architecture for scheduling demultiplexing requests and multiplexing replies, events and errors among X clients.
- **Multithreaded Architecture** - The X server is a multithreaded process capable of exploiting the nature of the operating system by breaking jobs into multiple threads for the operating system and hardware to perform. True pre-emptive multitasking, multithreaded environments offer a high degree of power for the X server.

Today's X Servers

Workstations

- powerful enough to handle complex computing requirements.
- usually display local X clients and a small percentage of network (remote) X clients.

X Terminals

- Dumb terminals with graphics capability.
- Download X Server software from host.
- Less expensive than workstations - simpler to maintain.

PC X Servers

- integrate PC and remote application server access into one

common desktop.

- leverage existing PC investment and user skill sets (desktop manipulation and access)
- flexibility - local or remote window management at the user's preference.
- ease of use

Over the last several years the desktop has evolved from a productivity or user-centric environment to one focussed on centralized administration surrounded by the adaptation of Web protocols and a browser based user interface. The latest release of the X Window System from X.Org - X11R6.5.1 - has addressed the issues of integrating X applications and browsers enabling rapid deployment without re-coding and security.



Client-Server Design


[Downloads](#)
[News](#)

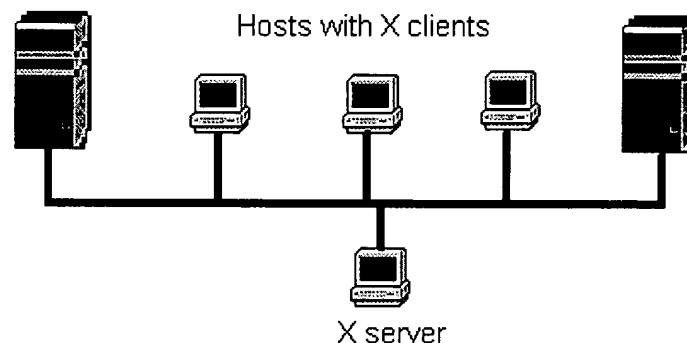

[X Protocol](#)
[Client-Server](#)
[Design](#)
[X Server Design](#)
[The Evolution of](#)
[Connectivity](#)
[Latest X Release](#)
[X Resources](#)
[Affiliated](#)
[Technologies](#)

[Calendar](#)
[Security](#)
[Bug Reports](#)
[X Help](#)
[Site Admin](#)

The design of the X Protocol specifies a client-server relationship between an application and its display. In X, the software that manages a single screen, keyboard and mouse is known as an X server. A client is an application that displays on the X server and is usually termed an X client or simply the application. The X client sends requests to the X server, for example a drawing or information request. The X server accepts requests from multiple clients and returns to the X client replies for information requests, user input and errors.

The X Server

- Runs on the local machine.
- Accepts and demultiplexes network (or local IPC) based X client requests and acts upon them.



The X server has seamless access to distributed applications.

The X server therefore:

- displays drawing requests on the screen.
- replies to information requests.
- reports an error in a request.
- Manages the keyboard, mouse and display device.
 - Multiplexes keyboard and mouse input onto the network (or via local IPC) to the respective X clients. (X events)
- creates, maps and destroys windows.
 - writes and draws in windows.

The X Client

Essentially an application written with the aid of libraries (i.e. Xlib, Xt) that take advantage of the X Protocol.

- sends requests to the server.
- receives events from server.

- receives errors from the server.

Protocol Messages

Requests

- X clients make requests to the X server for a certain action to take place. i.e.: Create Window
- To enhance performance, the X client normally does not expect nor wait for a response. The request is typically left to the reliable network layer to deliver.
- X requests are any multiple of 4 bytes.

Replies

- The X server will respond to certain X client requests that require a reply. As noted, not all requests require a reply.
- X replies are any multiple of 4 bytes with a minimum of 32 bytes.

Events

- The X server will forward to the X client an event that the application is expecting. This could include keyboard or mouse input. To minimize network traffic, only expected events are sent to X clients.
- X events are 32 bytes

Errors

- The X server will report errors in requests to the X client. Errors are like an event but are handled differently.
- X errors are the same size as events to simplify their handling. They are sent to the error handling routine of the X client. (32 bytes)